

REPRESENTATIONAL STATE TRANSFER (REST)

Michael Gfeller

- **Was ist REST**
- **ROA: Ressource Oriented Architecture**
- **Hateoas**
- **Versionierung**
- **Best Practices**

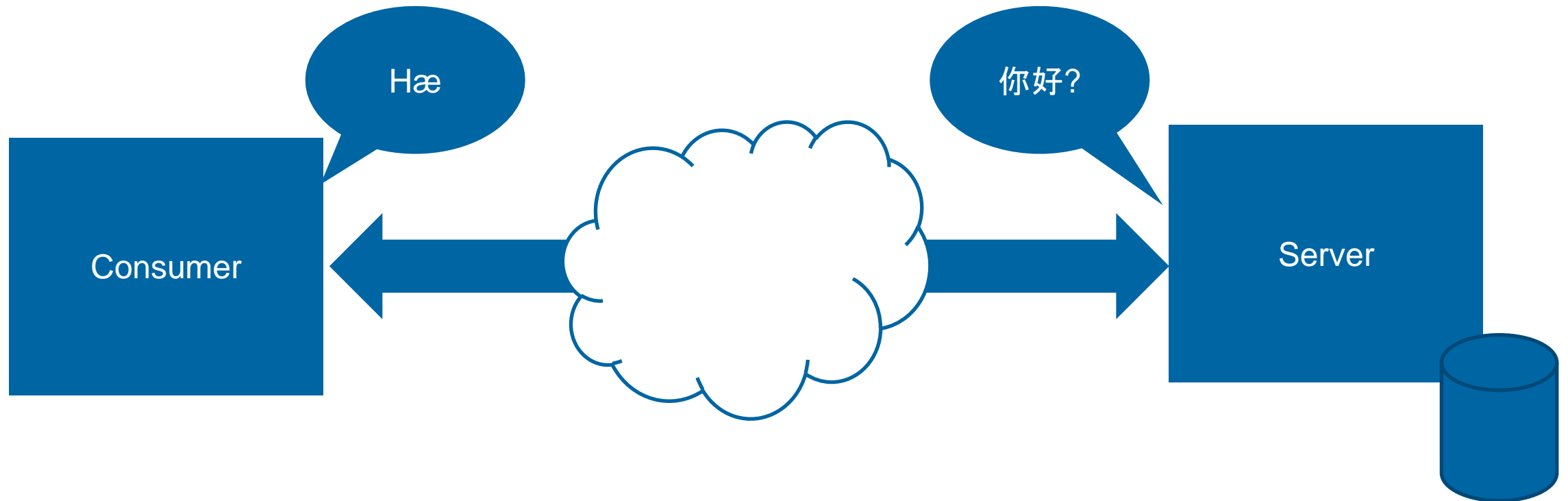
Die Teilnehmer...

- ... können den Einsatzzweck von REST erklären
- ... können die korrekte HTTP-Methode für eine Aktion auswählen
- ... kennen die Unterschiede der HTTP-Methoden im Context REST
- ... können das Konzept von Hateoas erklären
- ... kennen die Konzepte von REST und einer ROA und können diese an Beispielen erklären
- ... können erklären weshalb es Sinn macht Ressourcen zu verlinken

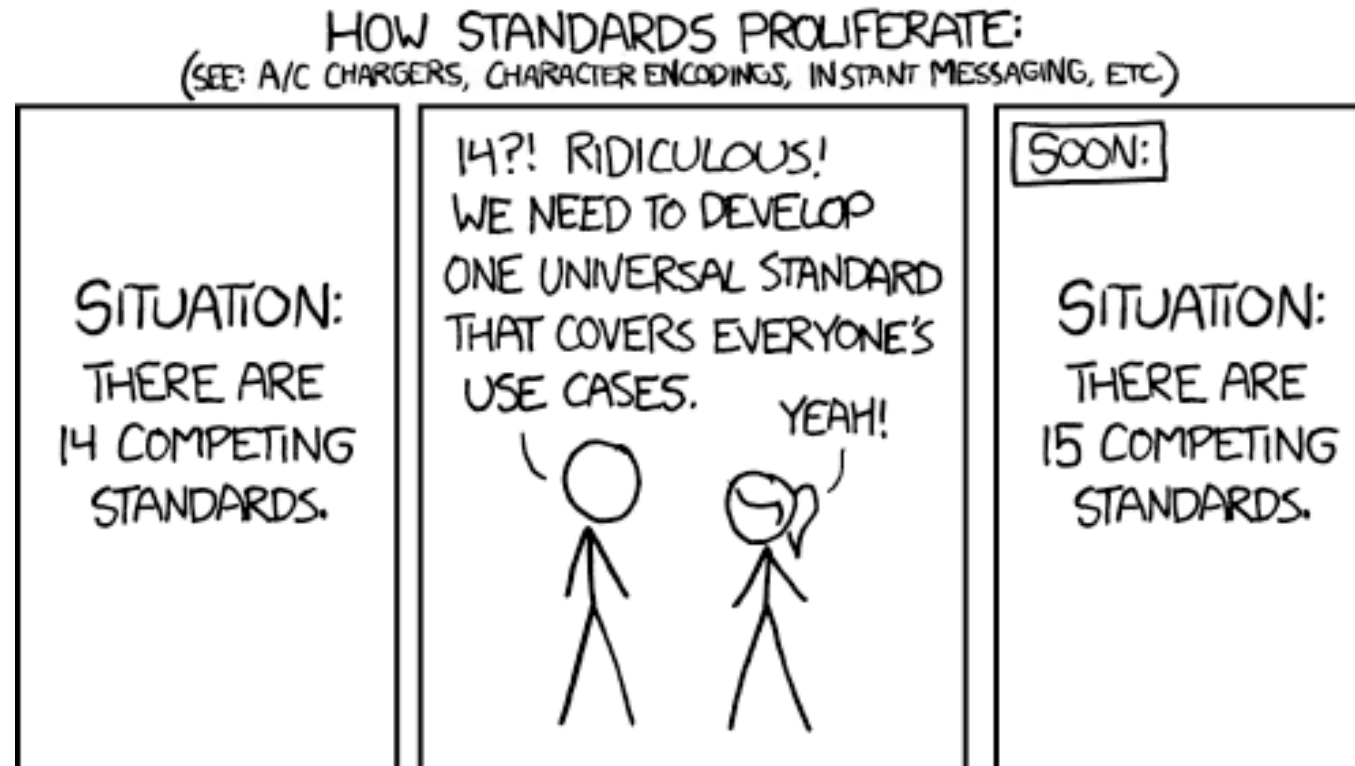
MOTIVATION

Motivation

- Ein Consumer möchte Daten von einem Server
- Consumer und Provider müssen die gleiche Sprache nutzen.



DIE WELT VOR REST



- **Viele verschiedene «Standards»**
 - RMI, RPC, Corba, DCE, DCOM, SOAP ...
- **Von vielen verschiedenen Organisationen**
 - Sun, Microsoft, IBM, OASIS, OMG
- **Mit vielen Problemen**
 - Schlechte Interoperabilität
 - Das Rad wurde immer wieder neu erfunden
 - Vendor «lock-in»

Die Welt vor REST: SOAP

```
<?xml version="1.0"?>
```

```
<soap:Envelope xmlns:soap=http://www.w3.org/2003/05/soap-envelope  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
```

```
<soap:Body>
```

```
  <m:GetPrice xmlns:m="https://www.shop.com/prices">
```

```
    <m:Item>Apples</m:Item>
```

```
  </m:GetPrice>
```

```
</soap:Body>
```

```
</soap:Envelope>
```

GET /Prices/Apple

```
<?xml version="1.0"?>
```

```
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
```

```
<soap:Body>
```

```
  <m:GetPriceResponse xmlns:m="https://www.shop.com/prices">
```

```
    <m:Price>1.90</m:Price>
```

```
  </m:GetPriceResponse>
```

```
</soap:Body>
```

```
</soap:Envelope>
```

{ price : 1.90 }

REST

*“Representational State Transfer (REST) is a software architecture style consisting of **guidelines** and **best practices** for creating scalable web services”*

Quelle: Wikipedia

RESTful is the adjective to the artificial subject word REST (this is a RESTful service)

Erste REST Beispiele

GET /orders/1

GET /customers/12/orders/1

DELETE /customers/12/orders/1

PATCH /orders/1

BODY: { **state** : "in progress" }

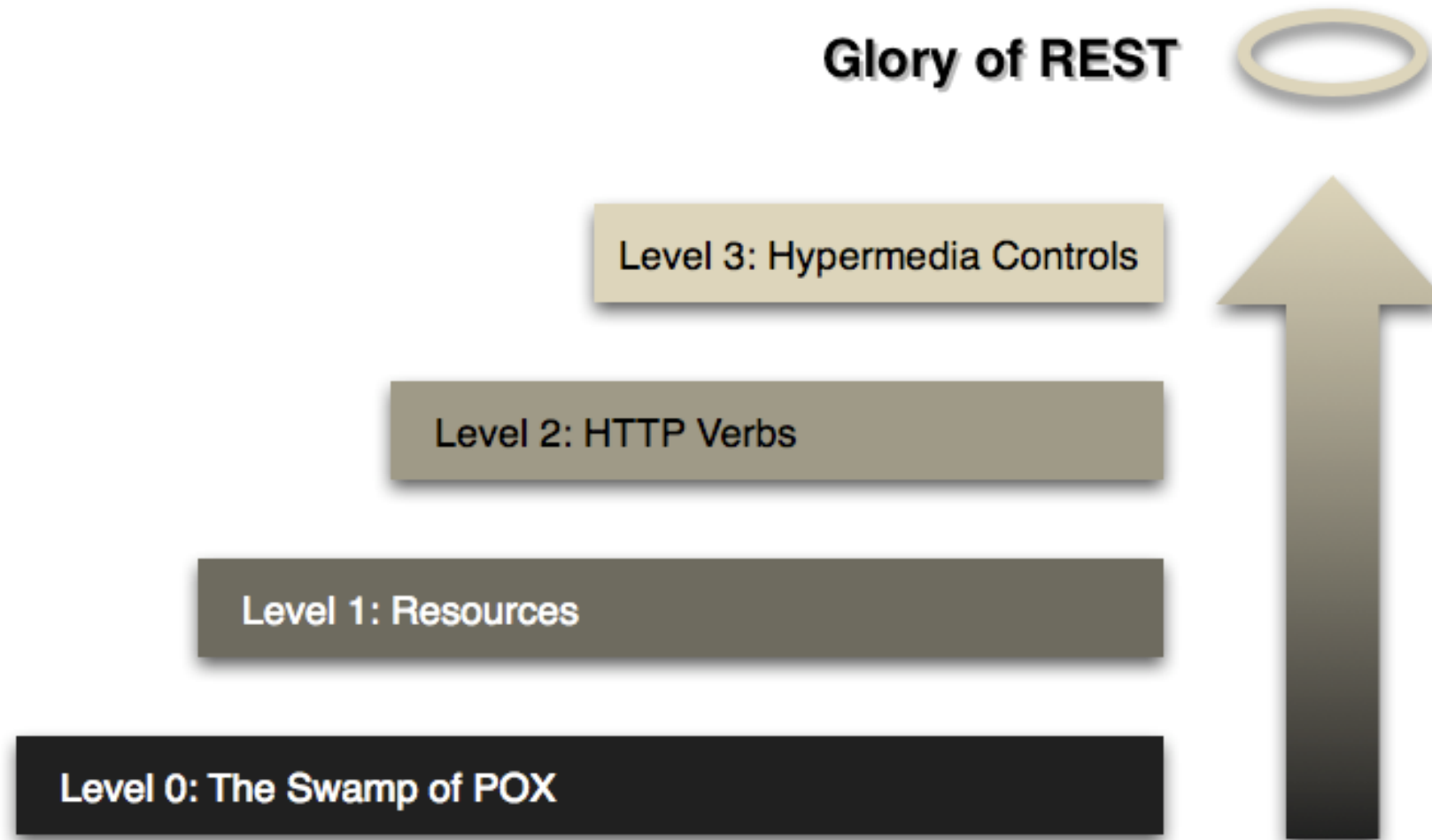
POST /orders

BODY: { productId : 33, amount: 10 }

■ **Verständlich?**

■ **Selbstbeschreibend?**

Richardson's Maturity Model



<http://martinfowler.com/articles/richardsonMaturityModel.html>

REST: EIGENSCHAFTEN

■ Client / Server

- Trennung von Client- und Server-Logik

■ Statuslose Kommunikation

- Jeder Request beinhaltet alle benötigten Informationen

■ Cache'bar

- Clients können antworten «zwischenspeichern» – insofern dies erlaubt wurde

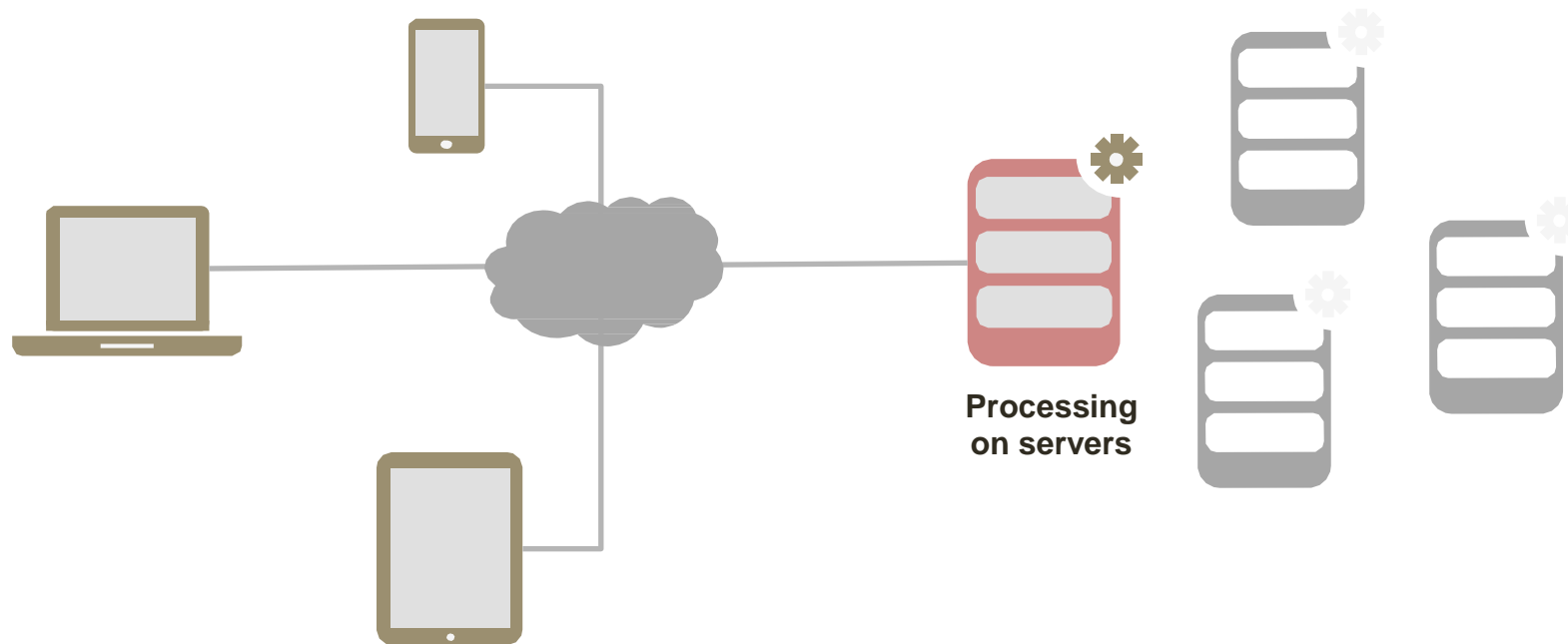
■ Layered System

- Erlaubt den Einsatz von Schichtenarchitekturen inkl. Load-Balancers, Proxies und Firewalls

■ Uniform Interface

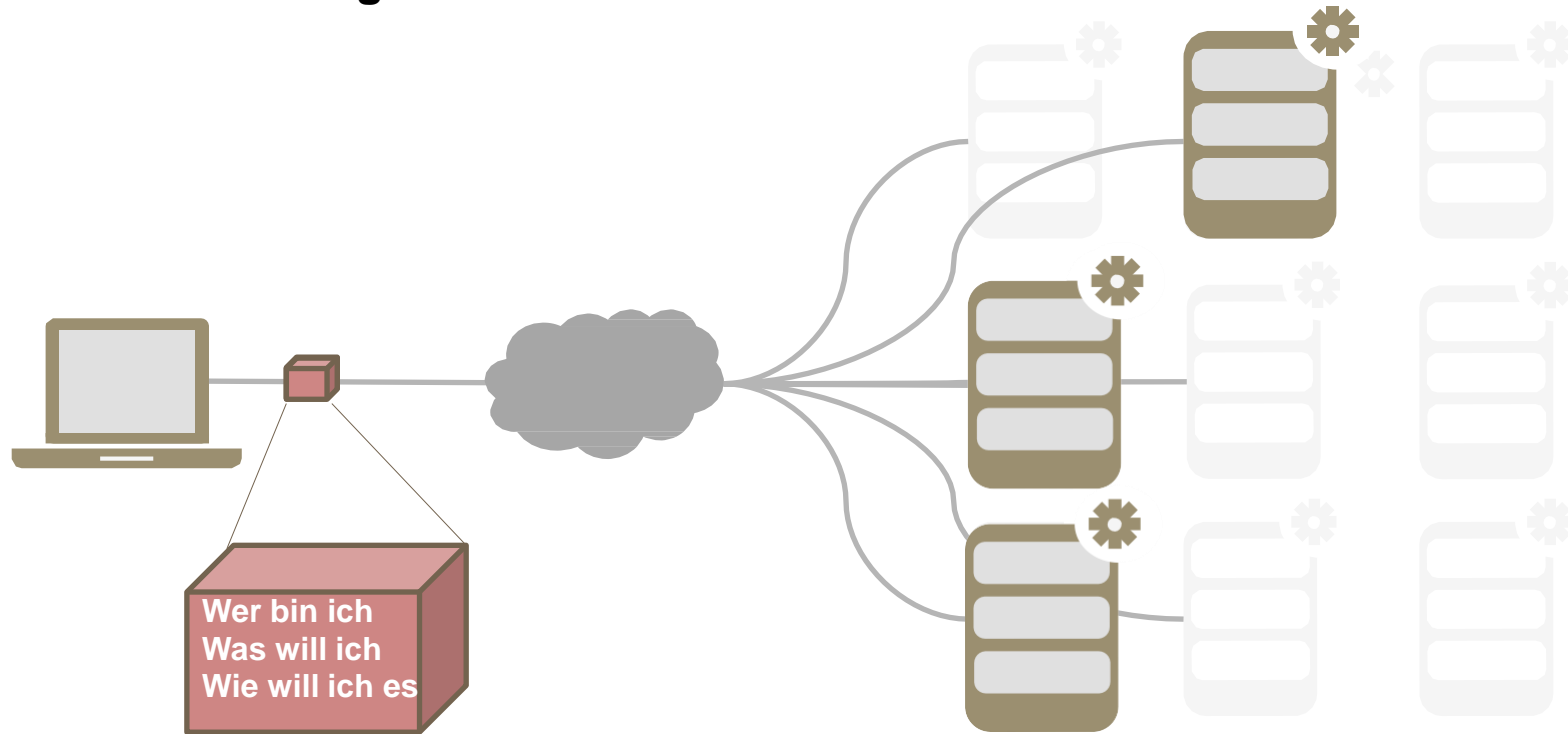
- Einheitliche Schnittstelle zw. Client & Server (Selbstbeschreibend)

■ Trennung von Client und Server-Logik



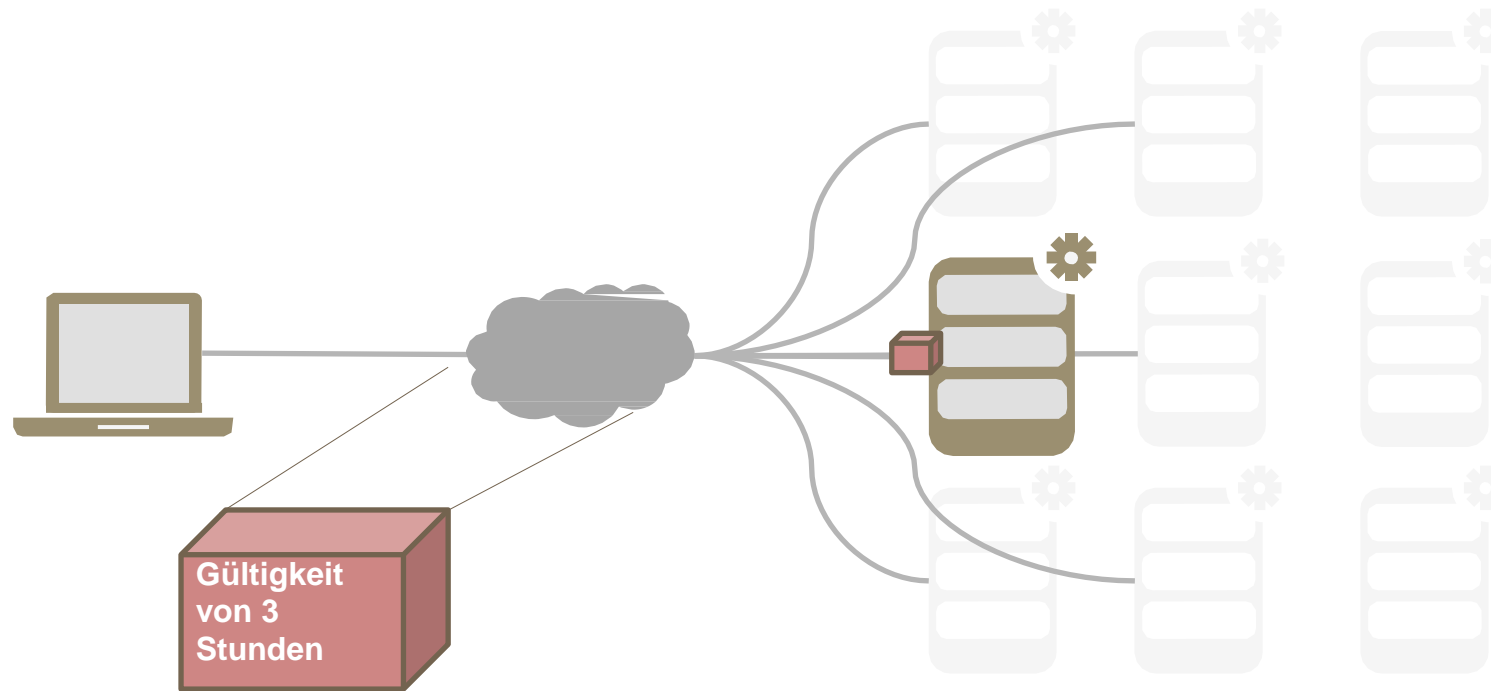
Eigenschaften von REST: Statuslose Kommunikation

- Jeder Request beinhaltet alle benötigten Informationen
- Ermöglicht hohe Skalierung



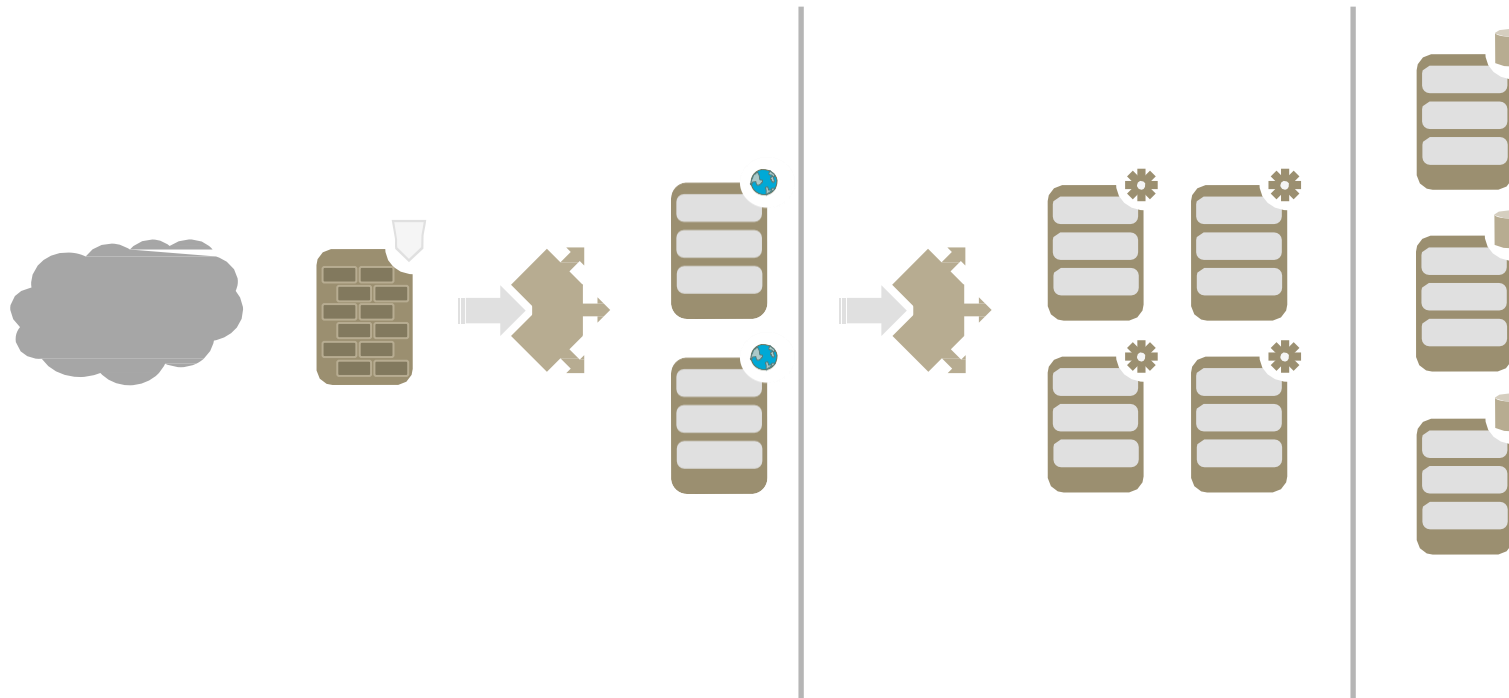
Eigenschaften von REST: Cache'bar

- Clients können Antworten zwischenspeichern – insofern dies erlaubt wurde.



Eigenschaften von REST: Layered System

- Erlaubt den Einsatz von Schichtenarchitekturen inkl. Load-Balancers, Proxies und Firewalls

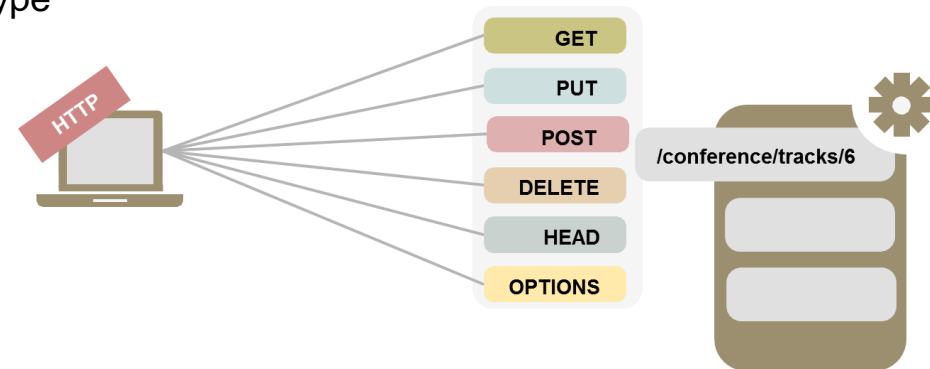


Eigenschaften von REST: Uniform Interface

■ Einheitliche Schnittstelle zw. Client & Server

■ Selbstbeschreibend

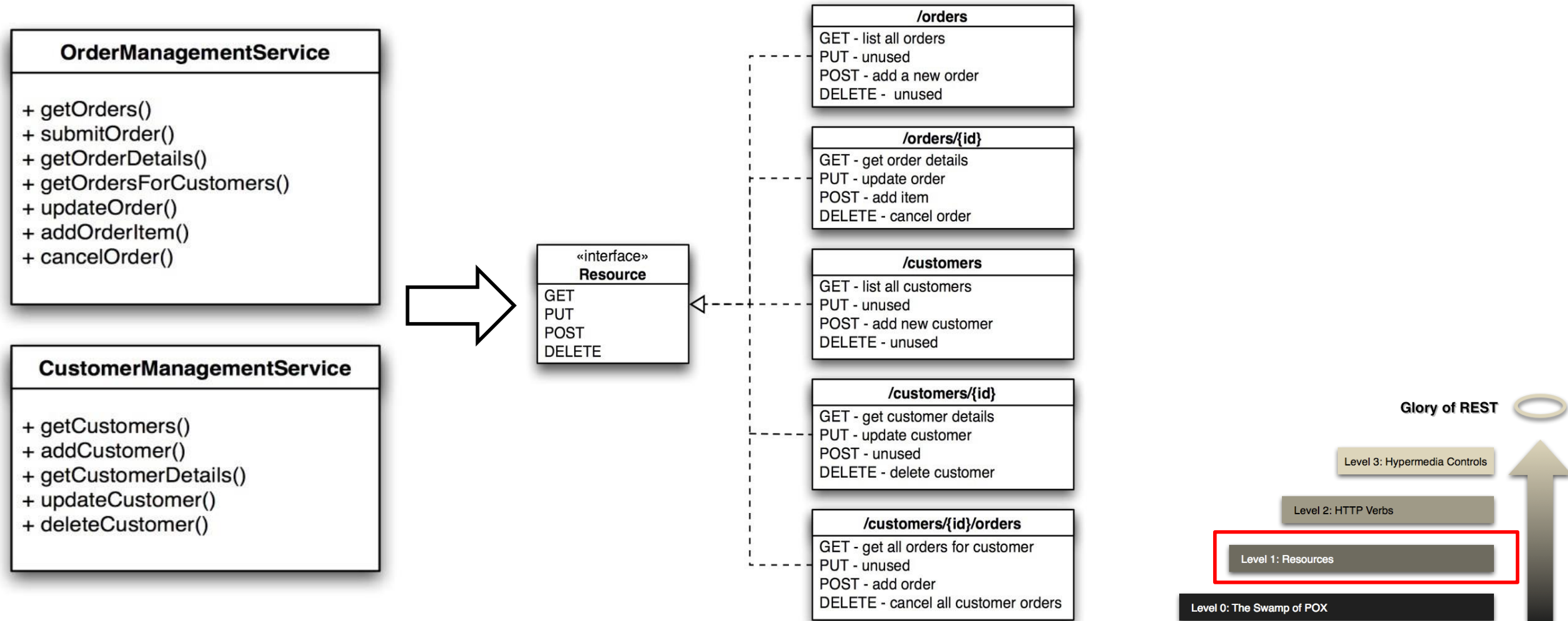
- Identification of resources
 - Die Ressource wird mit der Anfrage identifiziert. Z.B. URI
 - Die Ressource ist unabhängig von der Repräsentation welche zum Client geschickt wird.
 - Ermöglicht es unterschiedliche Formate zu wählen z.B. XML / JSON / HTML
- Manipulation of resources through representations
 - Die Antwort soll genügend Informationen beinhalten, um die Ressource ändern/löschen löschen zu können.
- Self-descriptive messages
 - Die Antwort muss beinhalten wie die Antwort zu behandeln ist. Z.B. MIME-Type
- Hypermedia as the engine of application state (HATEOAS)



RESOURCE ORIENTED ARCHITECTURE

Um was geht's?

■ Stelle die Ressource in den Mittelpunkt



■ Ressource

- Alles was genug wichtig ist um eigenständig referenziert zu werden.

■ Ressource Name

- Eindeutige ID der Ressource. Z.B. URI

■ Ressource Repräsentation

- «Dinge» haben mehrere Repräsentationen

■ Ressource Links

- Benutze Hyperlinks als Verknüpfung der «Dinge»

■ Ressource Interface

- Uniform Interface
- Benutze Standard-Methoden

■ Kommuniziere statuslos

- REST gibt Identifikationen als URL an.
- Jede Ressource hat eine eindeutige URL
 - orders/1
 - books/0-330-25864-8
- Sub-Ressourcen
 - Sub-Ressourcen verwenden wenn die Sub-Ressource ohne Parent nicht existieren kann. (Komposition)
 - Beispiele
 - customers/1/orders
 - topics/1/comments
 - Details
 - <https://stackoverflow.com/questions/13488697/restful-design-when-to-use-sub-resources>
 - <https://stackoverflow.com/questions/26795740/using-a-sub-resource-or-not>

■ URL-«Regeln»

- Ressourcen in Mehrzahl
- Ressourcen sind Nomen
 - /getAllCars => GET /cars
 - /createNewCar => POST /cars
 - /deleteAllRedCars => DELETE /cars?color=red
- Query-Parameter sind den Algorithmen / Filter vorbehalten z.B.:
 - Filtern: /orders?state=delay
 - Filtern: /movies?ranking gt 5&type eq action
 - Paging: /companies?skip=10&take=10
<http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api#pagination>
- ~~Bad-Example: /orders?id=10~~

ROA: Ressource Name: Typische Probleme

■ Filtern

- Teil der Ressource
orders/?state eq open

■ Sortierung

- Teil der Ressource
orders/?order-by=date&order-direction=asc

■ Suche

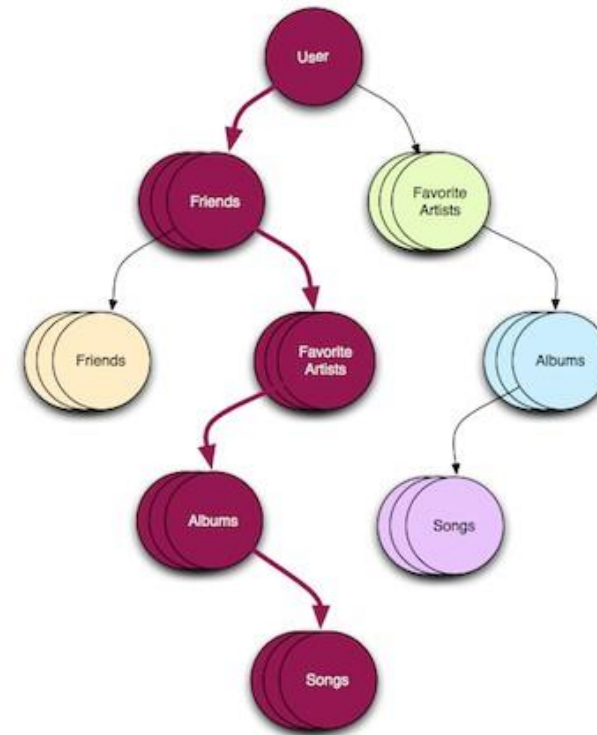
- Suche auf der Ressource ist üblicherweise ein Filtern
- (Full Text) Suche als eine Action
/search?....

■ Pagination

- GET /companies?skip=10&take=10
 - Üblicherweise werden Links mitgeschickt um eine Navigation zu ermöglichen
<http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api#pagination>

ROA: Ressource Links

```
1 {  
2   "username": "restKungFu",  
3   "birthdate": "26.06.1979",  
4   "friends": "http://api.myMusicStore.ch/users/restKungFu/friends/",  
5   "avatar": "http://api.myMusicStore.ch/users/restKungFu/avatar/320x250.jpg",  
6   "state": "registered",  
7   "account": {  
8     "mode": "premium",  
9     "plan": "monthly",  
10    "type": "creditcard"  
11  },  
12  "trusted": true  
13 }
```

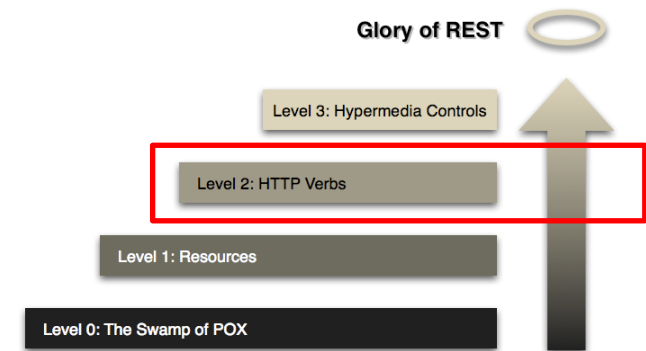


ROA: HTTP-VERBS

ROA: Benutze Standard-Methoden

- GET
- POST
- PUT
- DELETE
- HEAD
- OPTIONS
- PATCH

- <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>



GET

- Ressource wird angefordert (URI) = read
- Accept-Header definiert Repräsentation
- Query-Parameter sind lediglich Filter/Selektoren
- HTTP Status-Codes beachten

...

Request URL: <http://api.interhome.com/accommodations/CZ3940.210.1/?language=de&...20>
Accept: Application/json
Accept-Language: en-US,en
Host: api.interhome.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36
...

POST

- **Erzeugt eine neue Ressource**
 - Server ist führend für die URI- Generierung (im Gegensatz zu PUT)
- **Content-Type-Header angeben**
- **Wenn eine Ressource erzeugt wurde, wird im Response- Header «Location» die ID (URI) der Ressource mitgeteilt**
- **Status Codes: Conflict, Created, No Content, usw.**
- **POST's werden nie gecached**

POST api/orders/
BODY: { productId : 1234, amount: 2 }

Content-Type:application/json; charset=utf-8
Location: http://localhost:25211/api/orders/3

...

PUT

- **Überschreibt / erzeugt eine Ressource**
 - keine partiellen Updates.
- **URI wird von dem Client definiert (im Gegensatz zu POST)**
- **Status-Codes: Ok, No Content, Not Implemented, Created**
- **PUTs sind idempotent**
- **PUTs werden nie gecached**

PUT api/orders/3

BODY: { productId : 1234, amount: 2 }

POST vs. PUT

POST api/orders/

BODY: { productId : 1234, amount: 2 }

- Erzeugt eine neue Bestellung
- Server wählt die ID aus und returniert die erzeugte URL im Location-Header von der Response
 - Location: <http://localhost:25211/api/orders/3>

PUT api/orders/3

BODY: { productId : 1234, amount: 2 }

- Falls nicht vorhanden wird eine neue Bestellung erzeugt mit der id 3...
- ...falls vorhanden wird die vorhandene Bestellung komplett überschrieben – falls erlaubt

DELETE

- Löscht eine Ressource
- Status-Codes: Ok, Accepted, No Content
- DELETE's werden nie gecached

PATCH

- Wird für partielles Updaten einer Ressource genutzt.

PATCH /orders/1

Body:

```
{  
  status: "versendet"  
}
```

- Wichtig: Ein Patch muss atomar ablaufen. Ein GET Request sollte nie eine halbe Antwort erhalten.

OPTIONS

- Gibt an, wie die Ressource verwendet werden darf z.B. welche HTTP-Methoden erlaubt sind

...

Server: Apache/2.4.1 (Unix) OpenSSL/1.0.0g

Allow: GET,HEAD,POST,OPTIONS,TRACE

Content-Type: httpd/unix-directory

...

- Wird vom Chrome genutzt ob die CORS Headers gültig sind.

HEAD

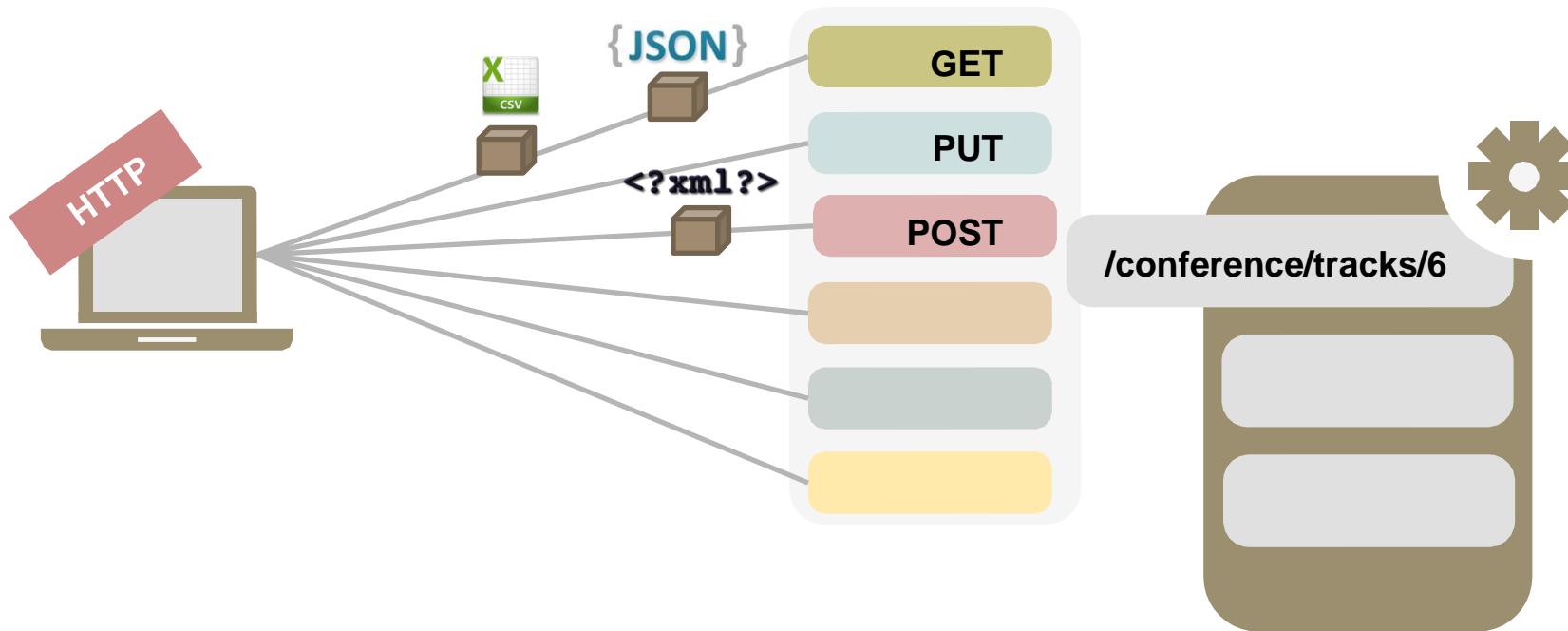
- Wird genau gleich angewendet wie GET, jedoch ohne die eigentliche Ressource zu erhalten
- Wird für Caching genutzt
- Header Informationen und Status-Codes sind relevant

ROA: FORMAT

Ressource Repräsentation

■ Gleiche Ressource, verschiedene Repräsentationen

- HTML, Json, XML, CSV, BusinessObjects, usw.
- Accept Request-Header definiert Repräsentation



Ressource Repräsentation

■ GET <http://api.interhome.com/services/status>

- Accept: application/json

```
{
  "serverName": "CH01S7A1",
  "status": "Available",
  "responseTime": 4,
  "processorUsageTotal": 72.95,
  "memoryTotal": 25769017344,
  "memoryUsageTotal": 14631698432,
  "processStatistics": {
    "id": 4120,
    "name": "w3wp",
    "handleCount": 8952,
    "threadCount": 305,
    "processorUsageTotal": 540.2881469726562,
    "processorUsageUser": 484.83917236328125,
    "processorUsagePrivileged": 54.54440689086914,
    "memoryWorkingSet": 11932078080,
    "memoryWorkingSetPrivate": 11833376768,
    "memoryWorkingSetPeak": 15336574976,
    "memoryVirtualBytes": 22293561344,
    "memoryVirtualBytesPeak": 22583402496
  },
  "systems": [
    ....
  ],
  "environment": "Production",
  "buildnumber": "IH_BusinessLayer_Build_20130702.3"
}
```

■ GET <http://api.interhome.com/services/status>

■ Accept: application/xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceStatusDto xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/Interhome.Models.Dto.Services.Status" >
  <ServerName>CH0157P7</ServerName>
  <Status>Available</Status>
  <Message i:nil="true" />
  <ResponseTime>12</ResponseTime>
  <ProcessorUsageTotal>58.46</ProcessorUsageTotal>
  <MemoryTotal>17151246336</MemoryTotal>
  <MemoryUsageTotal>15104229376</MemoryUsageTotal>
  <ProcessStatistics>
    <Id>2064</Id>
    <Name>w3wp</Name>
    <HandleCount>6562</HandleCount>
    <ThreadCount>356</ThreadCount>
    <ProcessorUsageTotal>693.97052001953125</ProcessorUsageTotal>
    <ProcessorUsageUser>607.60302734375</ProcessorUsageUser>
    <ProcessorUsagePrivileged>86.367515563964844</ProcessorUsagePrivileged>
    <MemoryWorkingSet>10701070336</MemoryWorkingSet>
    <MemoryWorkingSetPrivate>10602414080</MemoryWorkingSetPrivate>
    <MemoryWorkingSetPeak>10882011136</MemoryWorkingSetPeak>
    <MemoryVirtualBytes>17331048448</MemoryVirtualBytes>
    <MemoryVirtualBytesPeak>17398751232</MemoryVirtualBytesPeak>
  </ProcessStatistics>
  <Systems>
    </Systems>
    <Environment>Production</Environment>
    <Buildnumber>IH_BusinessLayer_Build_20130702.3</Buildnumber>
  </ServiceStatusDto>
```

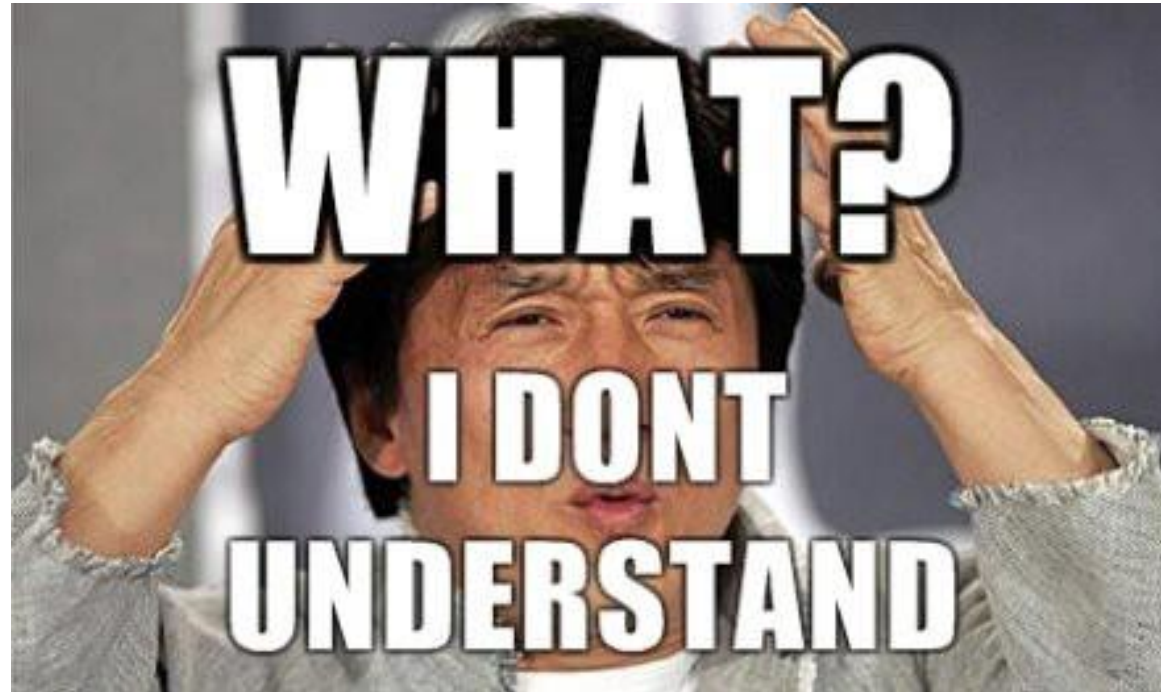
HATEOAS

HYPERMEDIA AS THE ENGINE OF APPLICATION STATE

HATEOAS (Aussprache: ha-TAY-oh-ahss)

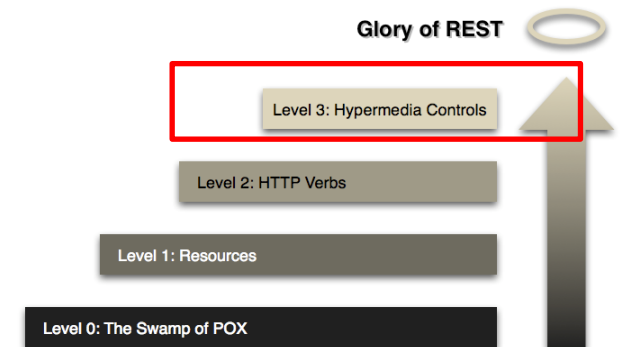
“The next control state of an application resides in the representation of the first requested resource, ... The application state is controlled and stored by the user agent ... anticipate changes to that state (e.g., link maps and prefetching of representations) ... The model application is therefore an engine that moves from one state to the next by examining and choosing from among the alternative state transitions in the current set of representations.”

■ Roy T. Fielding

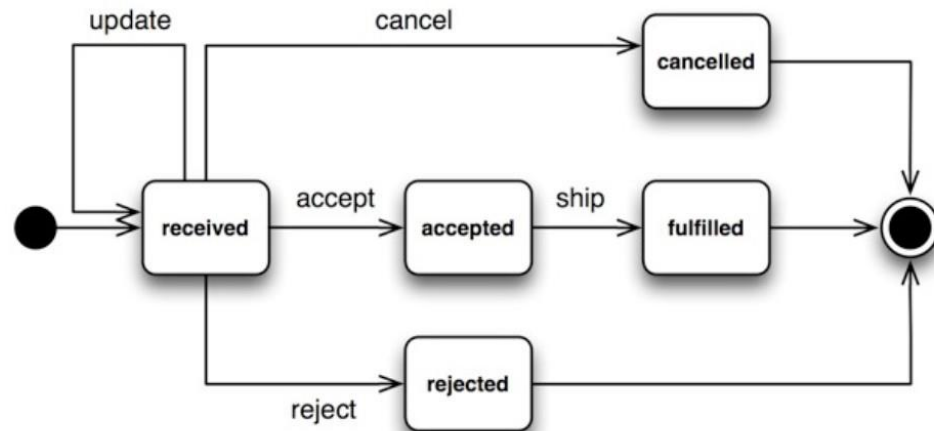


■ Hypermedia As The Engine Of Application State

- Prozessgedanke in der Ressource
- Media-Typen beschreiben die Ressource
- Aktionen werden ausgeführt beim folgen von Links
- Jede Antwort beinhaltet den «Application State»
- Selbstbeschreibende API's erzeugen Flexibilität
- Clients können die API «erforschen» ohne Dokumentation und Anleitung



HATEOAS



```
{
  "order": {
    "state" : "received",
    "...": "...",
    "links" : [
      {"rel" : "cancel", "href" : "http://..."},
      {"rel" : "accept", "href" : "http://..."},
      {"rel" : "reject", "href" : "http://..."},
      ...
    ]
  }
}
```

```
<reservations>
  <reservation id="928374" hotel-id="9876" price="200" date="10/05/2016" duration="5"/>
  <reservation id="1897" hotel-id="1234" price="60" date="10/05/2017" duration="2"/>
</reservations>
```

```
<reservations>
  <reservation id="928374" hotel="http://example/hotels/9876" price="200" date="10/05/2016" duration="5">
    <links>
      <link rel="self" href='http://example/customer/8888/reservations/928374' method='get'/>
    </links>
  </reservation>
  <reservation id="1897" hotel="http://example/hotels/1234" price="60" date="10/05/2017" duration="2">
    <links>
      <link rel="self" href='http://example/customer/8888/reservations/1897' method='get'/>
      <link rel="storno" href='http://example/customer/8888/reservations/1897' method='delete'/>
    </links>
  </reservation>
  <links>
    <link rel="self" href='http://example/customer/8888/reservations' method='get'/>
  </links>
</reservations>
```

■ Beispiel APIs:

- PayPal: <https://developer.paypal.com/docs/api/>
- Twitter: <https://dev.twitter.com/rest/>
- eBay: <http://developer.ebay.com/devzone/shopping/docs/CallRef/index.html>
- Sharepoint: <https://msdn.microsoft.com/en-us/library/office/jj860569.aspx>
- stripe.com: <https://stripe.com/docs/api>
- github.com: <https://developer.github.com/v3/>

■ Vorteile beim Einsatz von Hateoas

- Inline Dokumentation
- Explorable API
- Einfachere Clients, die URI ist sicher korrekt und aktuell.
- URI kann, falls gewollt, Serverseitig einfach geändert werden.

■ Nachteil

- (Sehr) aufwändig auf der Server-Seite

BEST PRACTICES

- <http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>
- <http://blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/>
- <https://github.com/Microsoft/api-guidelines/blob/master/Guidelines.md>
- **Wichtigste Punkte**
 - Use nouns but no verbs
 - Use plural nouns
 - Use HTTP status codes
 - Respect the meaning of the HTTP methods
 - GET method and query parameters should not alter the state
 - PUT is Idempotent, POST is not Idempotent
 - ...

- Die meisten Guidelines widersprechen sich in einigen Punkten.
- Die meisten Widersprüche basieren auf der Diskussion wie “Restful” eine Lösung ist.
 - Beispiel: <https://www.troyhunt.com/your-api-versioning-is-wrong-which-is/>

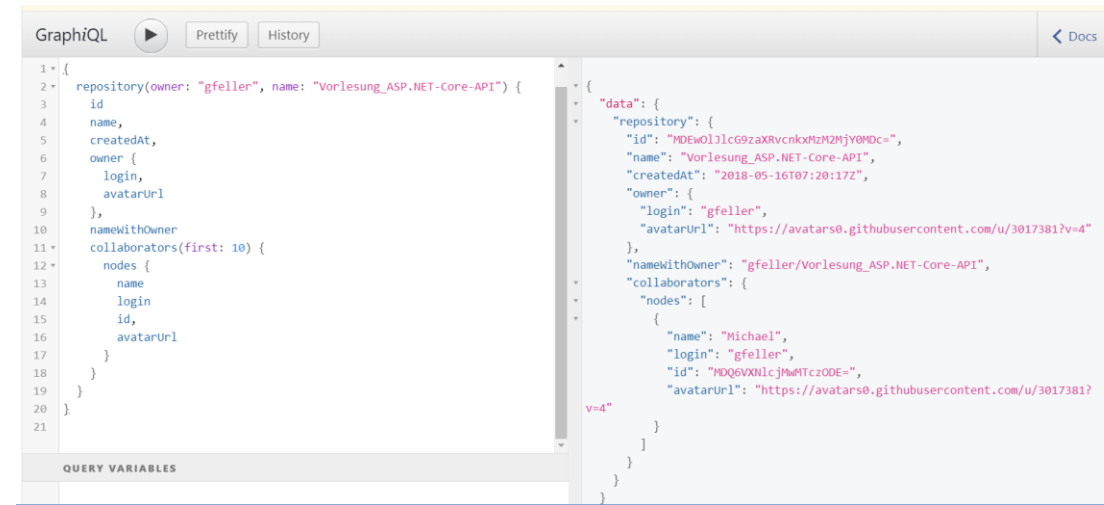


- Wichtig:
 - Entscheide Begründung und konsequent anwenden
 - API und Änderungen gut dokumentiert

EINSCHUB: GraphQL

GraphQL

- **GraphQL** ist eine alternative zu einer REST
- Ziele von GraphQL
 - Ask for what you need, get exactly that
 - Get many resources in a single request
 - Describe what's possible with a type system
 - ...
- Beispiel: <https://developer.github.com/v4/explorer/>



The screenshot shows the GraphQL Explorer interface. On the left, a query is written in a monospace font:

```
1 {  
2   repository(owner: "gfeller", name: "Vorlesung_ASP.NET-Core-API") {  
3     id  
4     name  
5     createdAt  
6     owner {  
7       login  
8       avatarUrl  
9     },  
10    nameWithOwner  
11    collaborators(first: 10) {  
12      nodes {  
13        name  
14        login  
15        id  
16        avatarUrl  
17      }  
18    }  
19  }  
20 }  
21
```

Below the query is a section labeled "QUERY VARIABLES". On the right, the JSON response is displayed:

```
{  
  "data": {  
    "repository": {  
      "id": "MDEwO1l1cG9zaXRvcnkxM2M2hjY0M0c=",  
      "name": "Vorlesung_ASP.NET-Core-API",  
      "createdAt": "2018-05-16T07:20:17Z",  
      "owner": {  
        "login": "gfeller",  
        "avatarUrl": "https://avatars0.githubusercontent.com/u/3017381?v=4"  
      },  
      "nameWithOwner": "gfeller/Vorlesung_ASP.NET-Core-API",  
      "collaborators": {  
        "nodes": [  
          {  
            "name": "Michael",  
            "login": "gfeller",  
            "id": "MDQ6VXNlcjMwMTczODE=",  
            "avatarUrl": "https://avatars0.githubusercontent.com/u/3017381?v=4"  
          }  
        ]  
      }  
    }  
  }  
}
```